International Journal of Grid Distribution Computing
Vol. 8, No.6, (2015), pp.171-184
http://dx.doi.org/10.14257/ijgdc.2015.8.6.17

Parallel Distributed Acceleration Based on MPI and OpenMP Technology

Feng Liu[1,2,3], Haitao Wu[1,3], Xiaochun Lu[1,3] and Xiyang Liu[4]

[1]*National Time Service Center, Chinese Academy of Sciences,*
[3]*East Shuyuan Road, Xi'an 710600, China*
[2]*University of Chinese Academy of Sciences,*
19A Yuquan Road, Beijing 100049, China
[3]*Key Laboratory of Precision Navigation and Timing Technology,*
Chinese Academy of Sciences, 3 East Shuyuan Road,
Xi'an 710600, China
[4]*Xidian University, 2 South Taibai Road, Xi'an 710071, China*
liufeng@ntsc.ac.cn, haitao@ntsc.ac.cn

Abstract

In order to speed up data processing in a signal monitoring and evaluation system, we need to use a parallel method. It is obvious that the traditional stand-alone store has no ability to satisfy the performance requirements, and the use of single core CPU is unable to content the severe requirement of speed. Consequently, multi-machine parallel acceleration technique based on MPI (cooperated with multi-core parallel acceleration technique based on OpenMP) can effectively solve all above problems. In this paper, a parallel distributed acceleration framework based on MPI and Open MP technology was given. Experimental tests were carried to verify our proposal. Finally, some suggestions to speed up the data processing was given.

Keywords: *Parallel distributed; MPI; OpenMP*

1. Introduction

Distributed computing studies how to put a problem which needs enormous computing power to solve into many small parts, then assigns these parts to many computers for processing, and finally obtains a final result by getting together the calculations results.

Multicore multiprocessor computers have spread around the world, as well as providing a lot of convenience to people's lives. However, with the development of society, relying solely on a computer processor has failed to meet the time, efficiency, and performance requirements, even if the computing capacity reaches a limited state.

Some measurements are needed in order to be proposed to overcome this problem. The first option is the use of the modern distributed parallel framework. This parallel framework can fully utilize the multi-processor and idle computing power. It transports data by the Internet and processes a variety of tasks in parallel.

The data in a spatial signal quality monitoring and evaluation system has characteristics such as large density, large number of files, a huge amount of data needs to be stored and processed. Traditional stand-alone storage and serial data processing can't meet the performance's requirements. So it is necessary to take into account the use of multi-machine parallel processing to speed up data processing, in order to further enhance the efficiency and performance

Now the parallel and distributed methods mainly cloud computing, GPU &CUDA, MPI & OpenMP and so on.

ISSN: 2005-4262 IJGDC

Cloud computing is a product of traditional computer and network technology integration. It includes distributed computing, parallel computing, utility computing, network storage, virtualization, load balancing, *etc*.

GPU & CUDA is a new computing paradigm. It makes full use of the advantages of CPU and GPU to achieve parallel and distributed computing.

MPI is a parallel library rather than a language. MPI is the representation of the standard or specification. It doesn't indicate one of the specific implementation. MPI is a message passing programming model and it becomes representative of this programming model. MPI is also a widely popular programming platform on cluster computing[1].

Cloud computing needs to build NFS platform, there are problems such as data security and data processing delay problems. GPU&CUDA programming model requires a computer with NVIDIA CUDA processor, but the general computer doesn't have this processor. MPI combines with FORTRAN77/C/Fortran90/C++, MPI doesn't change a serial language, only providing parallel languages callable library for serial language, so that the smallest changes to the original serial program, it's more convenient to use.

In summary, MPI-based distributed parallel acceleration technology has smaller overhead and is easy to implement [2]. However, in many cases, the use of pure MPI message passing programming model can't obtain the desired performance on such a multi-processor cluster configured. In order to combine the advantages of distributed memory with shared memory, people propose a distributed / shared memory hierarchy. OpenMP is a practical industry standard of shared memory programming. It's more widely to use MPI & OpenMP to achieve distributed / shared memory hierarchy. Hybrid programming model can take full advantage of two programming models: MPI can solve the communication of coarse-grained multi-processor. The OpenMP provides lightweight threads and a good solution to interact with each multiprocessor computer inside various processors [3].Compared with pure MPI model, MPI & OpenMP hybrid model uses the fast access to shared memory instead of message passing in nodes. This can reduce the cost of communications [4]. This hybrid programming model provides parallel between nodes (*i. e.*, multi-machine), the node (ie, stand-alone) and operating systems. This hybrid programming model can take full advantage of shared memory models and message passing model. It can effectively improve the performance of the system [5].

The remainder of this paper is organized as follows: Section 2 introduces the parallel framework. Section 3 introduces the experimental principle. Then in Section 4, we conduct tests. Finally, Section 5 concludes.

2. Related Work

The receiver receives data from satellite transmission and then merges multiple receiver's files into a big document. The receiver produces a data file every day. The data that involved in data processing is the data when received under normal tracking state.

The pseudo range, carrier phase, carrier-to-noise ratio, Doppler in raw data files are text format. The datum is stored in columns, one item per second. The first line of the file is the stored content's storage format. The size of the observed data file is 2GB.

Calculate and get the singular value points, loss of lock points and jump size, then marking their UTC time by using the second week of the original observation data, detecting receiver jump second phenomenon. You can get singular value points, loss of lock points and jump size through the comparison of the adjacent section of the original observation data. You can get the jump second by reading the receiver weeks seconds count.

Excluding singular value, then replacing the singular value with the average singular values in the pro domain in raw observation data. Dealing pseudo-range,

carrier phase data in a fifth-order polynomial fitting, getting a fitting curves by fitting raw data of ten minutes, getting the fitting difference of pseudo-range and carrier phase through using the normal data to divide fitting data, calculating the mean standard deviation. Then count standard deviation of the day, calculate the mean standard deviation.

Calculating B1, B2, B3 IQ slips pseudo range consistency, getting IQ slip pseudo range consistency by making a difference with raw data of IQ slip pseudo range, obtaining the final result by subtracting the mean consistency data. Then count standard deviation of the day, calculate the mean standard deviation.

Calculating ionospheric delay correction value and correcting pseudo-range measurements by using dual-frequency ionospheric formula, calculating B1I and B2I frequency, B2Q with B3Q frequency pseudo range consistency.

The use of dual-frequency ionospheric delay calculated correction values and correct pseudo-range measurements to calculate the pseudo B1I and B2I frequency, B2Q with B3Q frequency distance consistency. Pseudo-corrected pseudo-range measurements from the original concept of measuring the frequency plus ionospheric delay correction value, the pseudo B1I revised concept of pseudo-distance measurement and B2I corrected range measurements for the poor, pseudo B2Q corrected and pseudo-range measurements B3Q corrected for deviation from the observations, while subtracting the mean of the evaluation, that the two groups are asking to get the consistency of the final result of the pseudo-range, calculate the standard deviation of daily data and calculated results mean. Original pseudo-range measurements plus the frequency ionospheric delay correction value is the corrected pseudo-range measurements, the corrected pseudo-range measurements of B1I subtracts the corrected pseudo-range measurements of B2I、the corrected pseudo-range measurements of B2Q subtracts the corrected pseudo-range measurements of B3Q, subtracting the average value, then obtaining the required two groups pseudo range consistent final results. Calculating the standard deviation of the daily results and calculating the mean

The Calculated formulas are as the formula (1), (2)and (3):

$$\tilde{\rho}_i = \rho_i + \lambda_i^2 \frac{\Phi_j - \Phi_i}{\lambda_j^2 - \lambda_i^2} \tag{1}$$

$$\tilde{\rho}_j = \rho_j + \lambda_j^2 \frac{\Phi_i - \Phi_j}{\lambda_i^2 - \lambda_j^2} \tag{2}$$

$$\Delta\rho = \tilde{\rho}_i - \tilde{\rho}_j \tag{3}$$

$\tilde{\rho}_i$, $\tilde{\rho}_j$, ρ_i , ρ_j represent no ionospheric error ranging code pseudorange and ionospheric error ranging code pseudorange, j represents the frequency unlike i. Φ_i and Φ_j represent units of distance phase observations, λ_i and λ_j are wavelengths， $\lambda_i^2 \frac{\Phi_j - \Phi_i}{\lambda_j^2 - \lambda_i^2}$ is ionospheric error.

2.1. The Principles of MPI

As shown in Figure 1, there are four processes that are identified as 0,1,2,3. The running machine is called tp5. As for the implementation of results, the program itself has only one MPI print statements, but because it starts four processes and each process print operation simultaneously, so the final results of the implementation have four print statements [6].

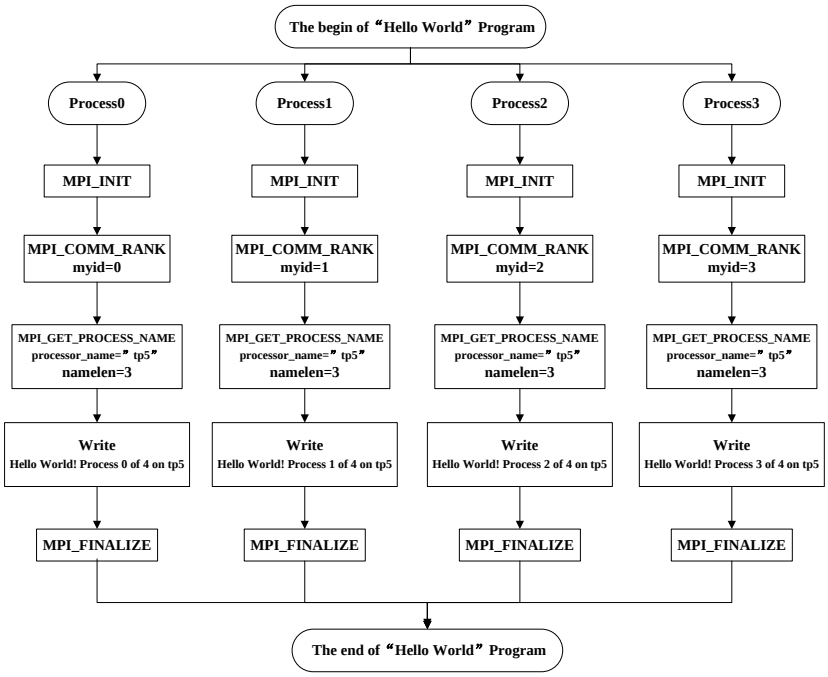

Figure 1. MPI Running Framework Example

2.2. The Principles of OpenMP

As shown in Figure 1, there are four processes that are identified as 0,1,2,3. The running machine is called tp5. As for the implementation of results, the program itself has only one MPI print statements, but because it starts four processes and each process print operation simultaneously, so the final results of the implementation have four print statements [6].

The implementation of OpenMP uses a Fork-Join model. The main thread creates and executes multiple threads according to the OpenMP compiler guidance statements when encountering parallel parts during execution, the number of threads created generally proportional to the number of the computer core [7].

As shown in Figure 2, the standard parallel mode's basic idea is as follows: There is only one main thread called Master Thread when the program begins running, the child threads execute serial program, the parallel parts derive other threads to execute; however, the serial parts can't be executed if the parallel parts aren't over. This is the standard parallel mode ---Fork/Join Parallel mode, the shared memory parallel programming also uses Fork / Join Parallel mode.

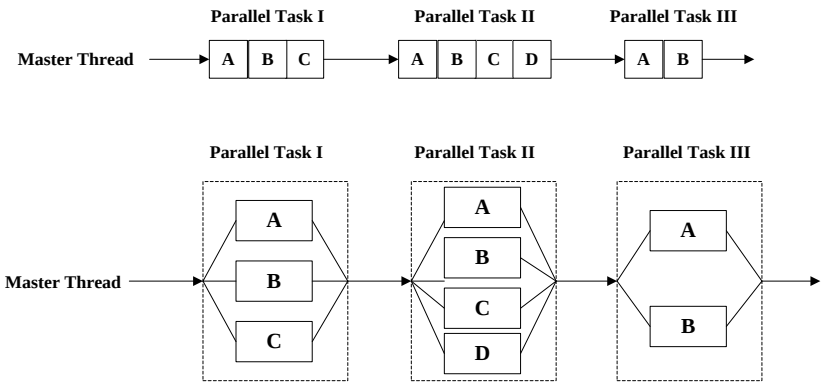

Figure 2. OpenMP Running Framework Example

3. Experiment Principle

Experimental machines are installed Ubuntu system, pre-installed MPICH2 and configured password ssh login authentication with multiple machines. Each machine using the same user name, then pre-transfer all copies of the data to every machine. The files have the same path.

3.1. Serial Proposal

As shown in Figure 3, the serial program is in accordance with the requirements of the needs analysis. Firstly, reading data line by line, then determining whether reading to the end, thirdly, getting the UTC time for Singular values, loss of lock points and jumping points, fourthly, removing singular value, UTC time of transition points, excluding the singular value , then polynomial fitting pseudo range and carrier phase, then obtaining the mean standard deviation, next pseudo range conformity assessment, then, correcting pseudo range ionospheric error data. Finally, outputting consumed time of the serial program, the running program ends successfully.

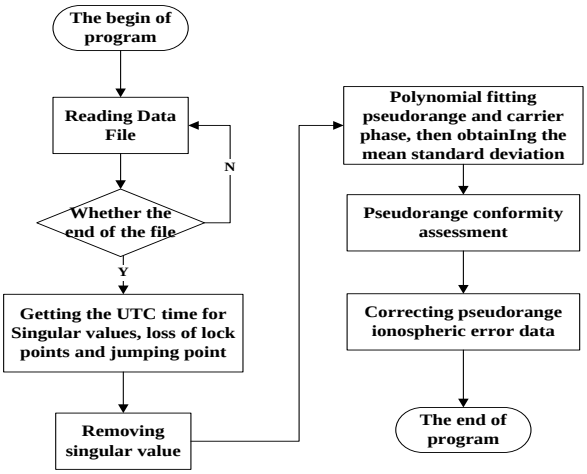

Figure 3. Serial Proposal Flow Chart

3.2. MPI & OpenMP Hybrid Parallel Proposal

For the MPI & OpenMP hybrid parallel proposal, assumptions used 5 computer parallel processing in Figure 4.

First of all, the main() function starts timing for memory mapping file, and then give the 5 computer tasks:

Process 0, in other words, is computer 0, it read serially, obtain locks, jump points and singular value of UTC time, eliminating singular values, assessing the pseudo range conformity, and then use OpenMP parallel guidance language to fix pseudo range ionospheric data in parallel; at the same time, using MPI_Recv () function to receive the finish task signal messages completed by the other 4 computers, statistical eventually run time; when a computer is 0 to receive all the computer to complete the task, calculate the entire program running time, and then finish the program.

There are 24 phases, 24 phases were allocated evenly to 4 machines.

Computer 1: the computer first using OpenMP parallel guidance statements to read B1 phase and pseudo range data in parallel, then run the polynomial fitting on each phase and calculating the standard deviation of average every day. When the

computer 1 finish assigned tasks, It will use MPI_Send () function to send a message to a computer 0 to notify it has completed the task.

Computer 2~4 work as same as Computer 1, Computer 2 processes only two B2 pseudo range, computer handles B3 pseudo range, Computer 4 processes B1, B2, B3's carrier data.

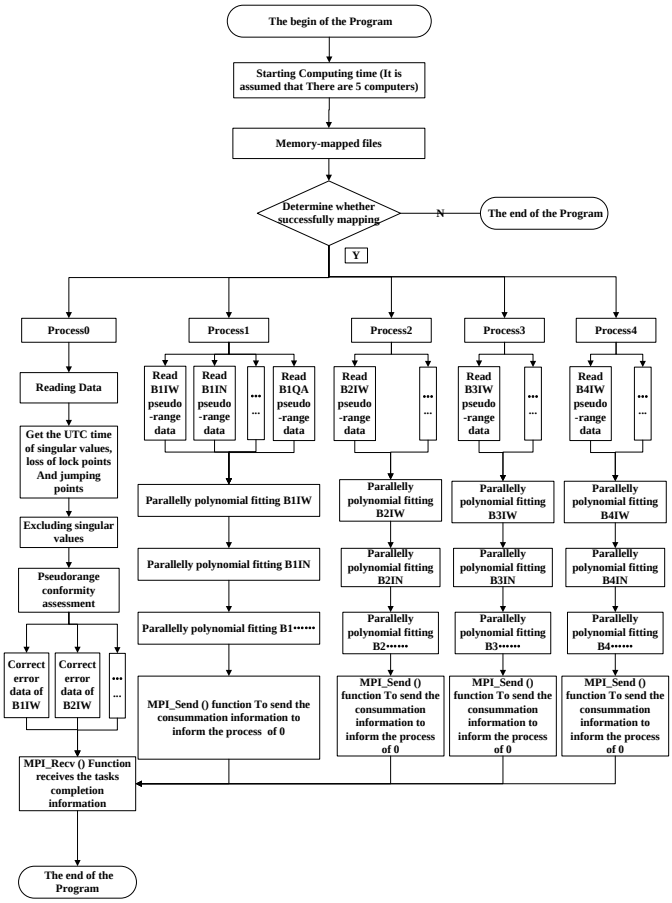

Figure 4. MPI & OpenMP Hybrid Parallel Proposal Flow Chart

3.3. The MPI Parallel Framework

For the MPI & OpenMP hybrid parallel proposal, assumptions used 5 computer parallel processing in Figure 4.

The MPI parallel framework allows users set the number of computers and the processes run on each computer on their demand. Figure 5 shows the MPI run the process in detail.

First of all, the main () function starts the MPI_Init () function, all the 5 computers will start doing their assigned tasks at this time, Figure 5 set computer 0 is the computer given priority to the other computers, receiving other four computer to complete the task messages, finally, computer 0 run the MPI_Finalize () function over the whole application.

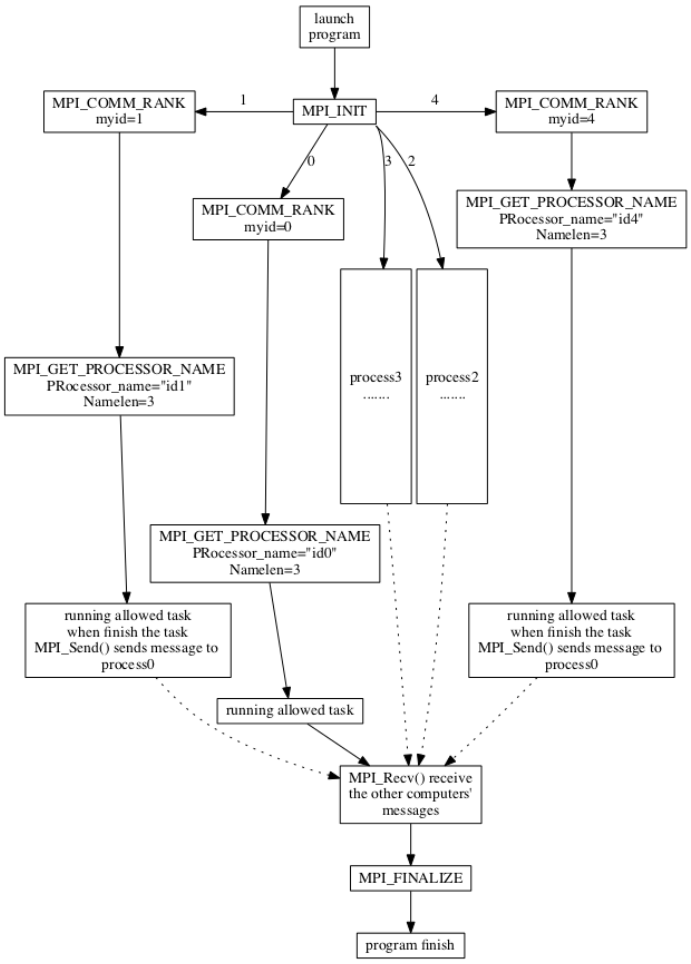

Figure 5. MPI Parallel Framework Flow Chart

3.4. Checking Data Parallelly

Computer 0 checks each phase data, due to the independence between 24 phase data. We can use OpenMP parallel block technology for each phase data. Figure 6 displays the flow chart of parallel correction.

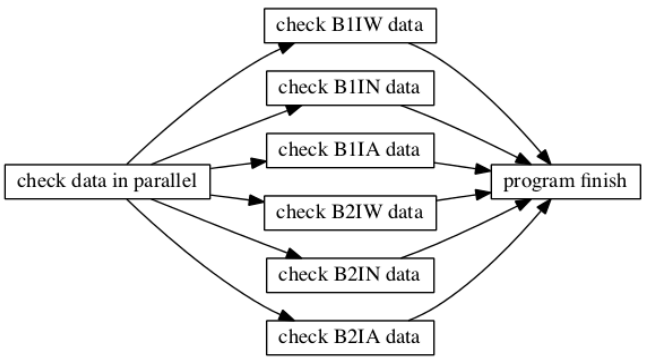

Figure 6. Checking Data Parallelly Flow Chart

3.5. Reading Data Parallelly

As shown in Figure 4, computer 1 ~ 4 respectively on the pseudo range and carrier phase polynomial fitting and calculate the mean standard deviation. Each of

the data is calculated by computer, to get so can use OpenMP parallel block technology on each computer parallel reading data.

Figure 7 shows the flow chart of parallel process of data read.

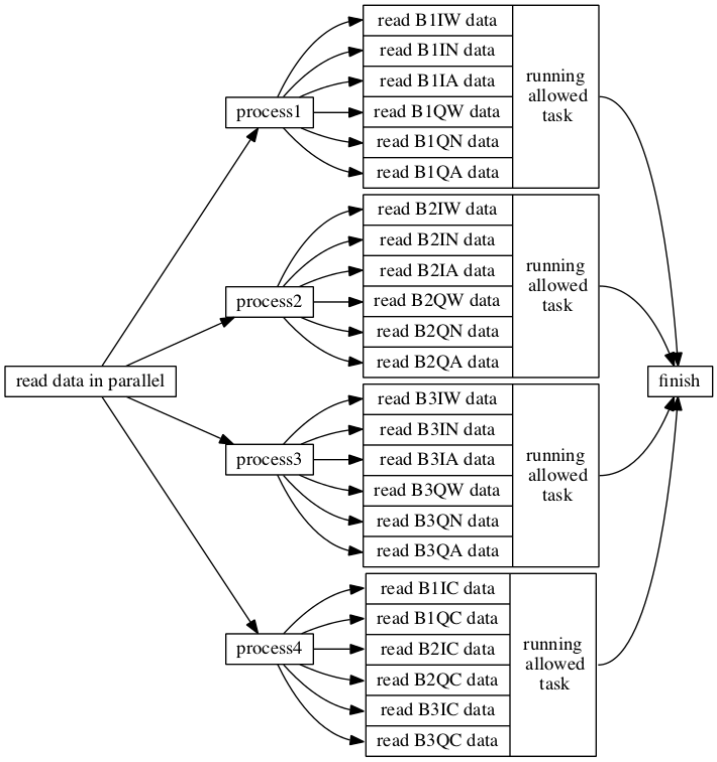

Figure 7. Reading Data Parallelly Flow Chart

3.6. Fitting Standard Deviation Parallelly

24 phase need to calculate daily average standard deviation of the carrier, the pseudo range. The solution is: once every ten minutes for fitting, take the normal data minus fitting data to get the pseudo range and carrier phase fitting difference, calculate the standard deviation, statistics the standard deviation calculation of standard deviation averages every day. Due to there is a total of 144 ten minutes a day. We realized it by using the POSIX thread to create 144 threads at the same time in a short period of time to deal with the standard deviation of every 10 minutes.

Figure 8 shows fitting process standard deviation parallel.

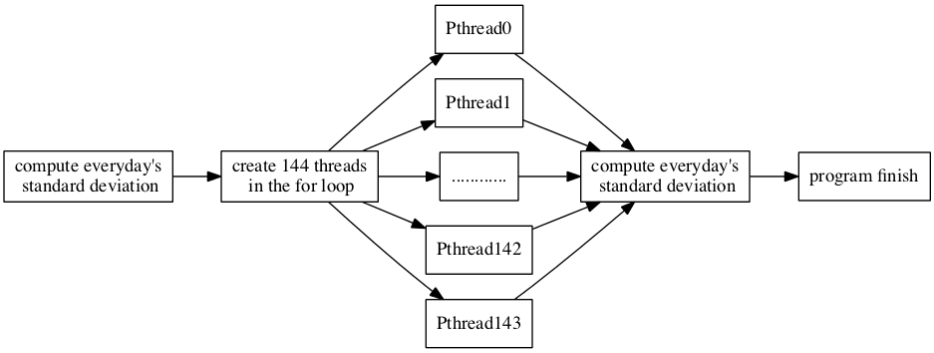

Figure 8. Fitting Standard Deviation Parallelly Flow Chart

4. Test

4.1. Test Plan

The test is divided into two models: a serial model, parallel MPI & OpenMP model.

MPI & OpenMP model respectively set up eight kinds of plans: two processes, the three processes, four processes, five processes, process of six, seven, eight, and nine.

The advantages of using MPI & OpenMP model: MPI can solve multiprocessor asked coarse-grained communication and OpenMP providing lightweight threads is a good way to solve every multiprocessor computer internal interaction between each processor. We can give the processor performance to the pole. In our plan, we have thought a lot about the MPI & OpenMP model.

In the MPI & OpenMP model, for the same kind of testing plan, think about of the amount according to the process and the performance of the processor, we assign different process a number of different tasks, to get the results of each test task. In order for the accuracy of the calculation results, we have to do computation on each task allocation 10 times, and then averaging. Picking out the optimal solution model in all test plans, the optimal solution is obtained the experiments the optimal solution.

Note: the default process program code 0 processing 24 phase data consistency check and correction, so according to the rules for the optimal solution is not all processes share the 24 tasks. All the test data are only for our experiment program. Experimental test was conducted on 3 computers, 3 computer configuration in Table 1.

Table 1. Experimental Environment Configuration

ID	Memory	Processor	IO Buffer	OS
Computer 1	3.9 GiB	Intel(R) Xeon(R) CPU W3505@2.53GHz × 2	4096 KB	32 bit ubuntu12.04 pae
Computer 2	5.7 GiB	Intel(R) Core(TM)i3 CPU M350@2.27GHz ×4	3072 KB	32 bit ubuntu11.10 pae
Computer 3	7.9 GiB	Intel(R) Xeon(R) CPU E5440@2.83GHz × 8	6144 KB	32 bit ubuntu12.04 pae

4.2. Serial Model

Serial model is all the process according to certain order serial computing, there is no parallel parts. Table 2 shows the time of average serial computing used.

Table 2. The Average Time Used in a Serial Model

Serial model (measuring unit: s)							
File Size	569M	898M	1.1G	1.5G	1.7G	2.0G	2.2G
Average time	48.849	71.27	80.083	110.589	123.008	151.611	162.229

4.3. Parallel Model

Due to many test data, this paper only shows the optimal data of each test plan. According to Table 3, we come to the conclusion:

Table 3. MPI & OpenMP Optimal Average Time on all Process Data with Serial Contrast

(P: process, measuring unit: s)

	2P	3P	4P	5P	6P	7P	8P	9P	1P
569M	21.3103	15.04	14.47	14.35	11.47	10.17	12.43	11.38	48.849
898M	33.6471	23.93	22.67	22.37	17.77	16.05	19.48	17.89	71.27
1.1G	39.4154	28.03	26.44	26.29	22.7	18.55	22.7	21.31	80.083
1.5G	57.6382	40.82	38.93	37.85	32.82	26.98	32.91	30.22	110.589
1.7G	63.5922	45.24	43.42	42.27	37.88	30.06	36.55	33.51	123.008
2.0G	79.2515	53.61	56.88	53.99	45.04	39.03	45.85	42.45	151.611
2.2G	85.4921	63.63	61.19	63.44	49.64	40.69	49.46	45.69	162.229

- The model of parallel running process of the optimal plan 7, relatively stable operation, the optimal overall performance.
- As a result of MPI in multimachine still exists between messaging overhead, such as can be seen from Figure 9 and not increase as the process operation results, the best process after the optimal value reaches a certain number operation result will decline.

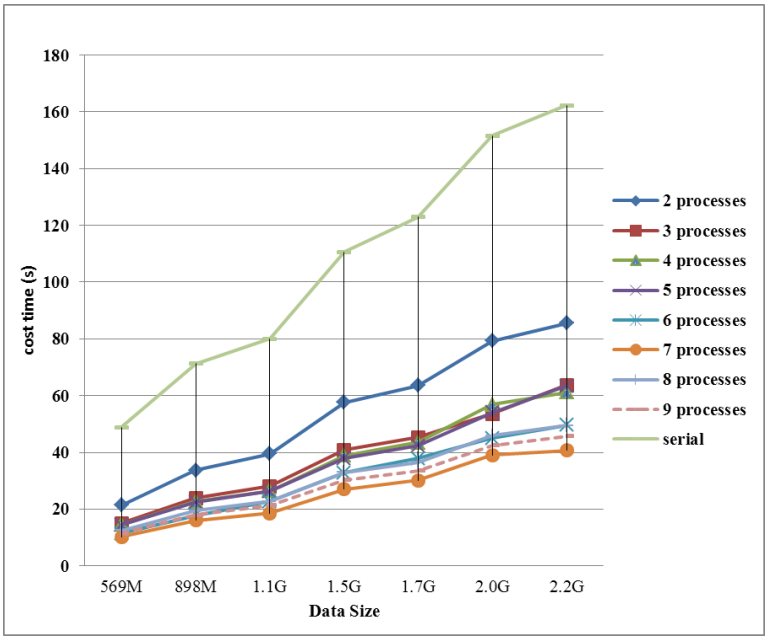

Figure 9. All Processes Optimal Solution Comparison Chart

- The experiment's optimal solution is just for our program, for the different needs of the project, we can determine the parallel model can reach a certain

speed, but the optimal parallel plan and the optimal cost time may not as same as our plan.

- Table 4 shows the pure MPI optimal solution and MPI & OpenMP optimal solution relative to a serial time reduced percentage.

Table 4. Parallel Optimal Schemes on Serial Reduction Percentage

	569M	898M	1.1G	1.5G	1.7G	2.0G	2.2G
Reduction percentage	79.18%	77.48%	76.84%	75.60%	75.56%	74.26%	74.92%

5. Conclusion

5.1. The Advantages of Parallel

The test is divided into two models: a serial model, parallel MPI & OpenMP model.

1) Improve the MPI Code Extensibility
The reason for hard to extend the MPI code is the balance the load of every machine. Using mixed programming patterns, can achieve better parallel granularity. MPI only responsible for the communication between nodes, a coarse-grained parallel; OpenMP implementation of internal parallel, because the OpenMP has no load balancing problems, to improve the performance.

2) Copies of Data
Data copies are often limited to memory, and owing to its poor scalability and global communication. Every node in pure MPI applications, the size of the memory is divided into the number of processors. And hybrid model can for the whole node memory processing, can achieve more ideal problem domain.

3) Convenient to Achieve Optimum Parallel
In some cases, the MPI application implementation performance does not improve with the increase of the number of processors, but there has an optimal value. When using the mixed programming model will be beneficial, because we can use OpenMP threads instead of processes that can reduce the number of processes required, to run the ideal number of MPI process, reoccupy OpenMP further decomposed task, making all processors run efficiently. Due to computer configuration, limits can no longer create OpenMP parallel area. We use the bottom of the POSIX thread to create lightweight threads to achieve further in parallel.

5.2. Summary

Parallel and distributed technology is very mature both at home and abroad, there are also many implementation frameworks, such as cloud computing, CUDA+GPU, and OpenMP+MPI, The reasons for our paper bases on the MPI + OpenMP technical are mainly the technology is very mature, in the parallel framework is easy to satisfy the high speed and large capacity of data processing. By using the parallel framework we get the following conclusion:

- Parallel design usually were better than the serial design;
- MPI & OpenMP model makes full use of advantages of MPI and OpenMP each, and each other at the same time they can make up for their shortcomings;
- All the parallel frameworks are not with the more the process number, the more speed, when to achieve an optimal value, along with the increase in the number of process, the process speed slows down instead;

- In the same configuration environment, different parallel plan program cost time difference is very big, and every kind of parallel plan has its own optimal solution;
- We try to use the same computer configuration on every test, so that we can minimize the performance problems caused by the different computer configuration, or low configuration computer cost time will become the bottleneck of the whole cost time.
- The experiment's optimal solution is just for our program, for the different needs of project, we can determine the parallel model can reach a certain speed, but the optimal parallel plan and the optimal cost time may not as same as our plan .The reason for this is the application of parallel granularity and so on.

References

[1] K. Hwang, "Advanced Computer Architecture: Parallelism", Scalability, Programmability, China Machine Press, Beijing **(1999)**.
[2] Y. Feng, S. Y. Zhou, "Research on Development of Mixed Mode MPI+OpenMP Applications", Computer System and Application. vol. 15, no. 2, **(2006)**.
[3] J. Q. Michael, "Parallel Programming in C with MPI and OpenMP", McGraw Hill Higher Education, New York **(2003)**.
[4]] S. G. Caglar, G. D. Benson, Q. Huang, C. W. Chu, "USFMPI: A Multi-threaded Implementation of MPI for Linux Clusters. Proceedings of the 15th International Conference on Parallel and Distributed Computing and Systems", Louisville, USA, **(2002)** September.
[5] Y. H. Zhao, X. B. Chi, "MPI+OpenMP Hybrid Paradigms and Efficient Implementation Base on SMP Clusters", Microelectronics and Computer. vol. 10, **(2005).**
[6] MPI: a message-passing interface standard. http://www.mpi-forum.org **(1995)**.
[7] OpenMP C and C++ Application Program Interface. http://www.openmp.org **(2002)**.

Authors

Feng Liu, he received his master's degree in Measuring and Testing Technologies and Instruments from Graduate School of Chinese Academy of Sciences, China, in 2008. Now he is studying the PhD of Astrometry and Celestial Mechanics in University of Chinese Academy of Sciences, China. His current research interests include the applications of cloud computing and parallel computing in GNSS.

Haitao Wu, he received his PhD in Astrometry and Celestial Mechanics from Graduate School of Chinese Academy of Sciences, China, in 2002. Now he is a research professor in National Time Service Center (NTSC), Chinese Academy of Sciences. His current research interests focus on overall technical and applications of satellite navigation system.

Xiaochun Lu, she received his PhD in Astrometry and Celestial Mechanics from Graduate School of Chinese Academy of Sciences, China, in 2004. Now she is a research professor in National Time Service Center (NTSC), Chinese Academy of Sciences. Her current research interests focus on satellite navigation signal design and assessment.

Xiyang Liu, he received his PhD in Circuits and Systems from Xidian University, China, in 2007. Now he is a professor in School of Software of Xidian University. His current research interests focus on distributed computing and software testing.